Lecture Notes:
Discrete Mathematics for Computer Science

Vladimir Lifschitz
University of Texas at Austin

Part 9. Annotated Programs

A Toy Programming Language

In our toy programming language, all variables are integer variables, so that there is no need to declare them. *Integer expressions* are formed from variables and integer constants using the operators $+$, $-$ and $\times$. *Boolean expressions* are formed from equalities and inequalities between integer expressions using the propositional connectives $\wedge$, $\vee$, $\neg$. *Assignments* are expressions of the form $V \leftarrow IE$ where V is a variable and IE is an integer expression. *Programs* are formed from assignments using three constructs:

- sequential composition

  $$P; Q$$

  where P and Q are programs,

- conditional composition

  if BE **then** P **else** Q **endif**

  where BE is a Boolean expression and P, Q are programs,

- loop

  while BE **do** P **enddo**

  where BE is a Boolean expression and P is a program.

For instance,

if $m > n$
  then $p \leftarrow m$
  else $p \leftarrow n$
endif

is a program. We will denote it by $P1$. Another example:

$p \leftarrow 1$;
$i \leftarrow 0$;
while $i < n$ **do**
  $i \leftarrow i + 1$;
  $p \leftarrow p \times 2$
enddo

We will call this program $P2$.

Partial Correctness and Annotated Programs

An *assertion* is a condition that may be true or false depending on the values of variables. For example, every Boolean expression is an assertion. An assertion may include symbols that are not allowed in programs. For instance, the equalities $p = \max(m, n)$ and $p = 2^n$ are assertions, even though the espressions $\max(m, n)$ and 2^n are not part of our programming language.

We will describe properties of programs using formulas of the form

$$\{A\}\ P\ \{B\} \tag{1}$$

where A and B are assertions and P is a program. Assertion A is called the *precondition* of formula (1), and B is called its *postcondition*. Formula (1) expresses that if the values of variables satisfy condition A then their new values after the execution of P will satisfy condition B, provided that the execution terminates. For instance, the formula

$$\{m = 3\}\ n \leftarrow m + 5\ \{m = 3 \wedge n = 8\} \tag{2}$$

is true. Formulas

$$\{m = 3 \wedge n = 5\}\ P1\ \{p = 5\} \tag{3}$$

and

$$\{n = 5\}\ P2\ \{p = 32\} \tag{4}$$

are true as well. On the other hand, formula

$$\{n = 5\}\ P1\ \{p = 5\}$$

is false. (Couterexample: before the execution of $P1$, $n = 5$ and $m = 6$.)

If formula (1) is true, we say that program P is *partially correct* for the precondition A and postcondition B.

An *annotated program* is a program with assertions, enclosed in braces, possibly added to its text at various points. For instance, every program is an annotated program (no assertions added). Every formula of the form (1) is an annotated program also (the precondition and postcondition are added). We define below rules for constructing "complete" annotated programs. If we can turn a formula (1) into a complete annotated program by inserting additional assertions between its precondition and postcondition then we can say that the formula is proved.

Substitution

The following notation will be useful. If A is a condition, V is a variable, and IE is an integer expression, then by A^V_{IE} we denote the condition obtained from A by substituting IE for V. For instance,

$$(m = 3 \wedge n = 8)^n_{m+5} \quad \text{is} \quad m = 3 \wedge m + 5 = 8.$$

If V occurs in A several times then A^V_{IE} is obtained from A by substituting IE for all occurrences of V simultaneously. For instance,

$$(m < n \wedge n = 8)^n_{n+1} \quad \text{is} \quad m < n + 1 \wedge n + 1 = 8.$$

If V doesn't occur in A at all then A^V_{IE} is the same as A:

$$(m = 3)^n_{m+5} \quad \text{is} \quad m = 3.$$

Rule of Assignment

Any annotated program of the form

$$\begin{array}{l} \{A^V_{IE}\} \\ V \leftarrow IE \\ \{A\} \end{array}$$

is complete.

For instance, the annotated program

$$\begin{array}{l} \{m = 3 \wedge m + 5 = 8\} \\ n \leftarrow m + 5 \\ \{m = 3 \wedge n = 8\} \end{array}$$

is complete.

Rules of Consequence

1. An annotated program of the form

$$\begin{array}{l} \{A\} \\ \{B\} \\ P \end{array}$$

is complete if A implies B and the annotated program

$$\begin{array}{l} \{B\} \\ P \end{array}$$

is complete.

2. An annotated program of the form

$$\begin{array}{l} P \\ \{A\} \\ \{B\} \end{array}$$

is complete if A implies B and the annotated program

$$\begin{array}{l} P \\ \{A\} \end{array}$$

is complete.

For instance, the annotated program

$$\begin{array}{l}\{m = 3\}\\ \{m = 3 \wedge m + 5 = 8\}\\ n \leftarrow m + 5\\ \{m = 3 \wedge n = 8\}\end{array}$$

is complete, because the assertion $m = 3$ implies $m = 3 \wedge m + 5 = 8$. This annotated program justifies formula (2).

Rule of Composition

An annotated program of the form

$$\begin{array}{l}P;\\ \{A\}\\ Q\end{array}$$

is complete if each of the annotated programs

$$\begin{array}{l}P\\ \{A\}\end{array}$$

and

$$\begin{array}{l}\{A\}\\ Q\end{array}$$

is complete.

For example, the annotated program

$$\begin{array}{l}\{m = 2\}\\ \{m + 1 = 3\}\\ m \leftarrow m + 1;\\ \{m = 3\}\\ \{m = 3 \wedge m + 5 = 8\}\\ n \leftarrow m + 5\\ \{m = 3 \wedge n = 8\}\end{array}$$

is complete. It justifies the formula

$$\begin{array}{l}\{m = 2\}\\ m \leftarrow m + 1;\\ n \leftarrow m + 5\\ \{m = 3 \wedge n = 8\}.\end{array}$$

Conditional Rule

An annotated program of the form

$$
\begin{array}{l}
\{A\} \\
\textbf{if } BE \\
\quad \textbf{then } \{A \wedge BE\} \\
\quad P \\
\quad \{B\} \\
\quad \textbf{else } \{A \wedge \neg BE\} \\
\quad Q \\
\quad \{B\} \\
\textbf{endif} \\
\{B\}
\end{array}
$$

is complete if the annotated programs

$$
\begin{array}{l}
\{A \wedge BE\} \\
P \\
\{B\}
\end{array}
$$

and

$$
\begin{array}{l}
\{A \wedge \neg BE\} \\
Q \\
\{B\}
\end{array}
$$

are complete.

For instance, the annotated program

$$
\begin{array}{l}
\{m = 3 \wedge n = 5\} \\
\textbf{if } m > n \\
\quad \textbf{then } \{m = 3 \wedge n = 5 \wedge m > n\} \\
\quad \{\mathit{false}\} \\
\quad \{m = 5\} \\
\quad p \leftarrow m \\
\quad \{p = 5\} \\
\quad \textbf{else } \{m = 3 \wedge n = 5 \wedge m \leq n\} \\
\quad \{n = 5\} \\
\quad p \leftarrow n \\
\quad \{p = 5\} \\
\textbf{endif} \\
\{p = 5\}
\end{array}
$$

is complete. It justifies formula (3).

While Rule

An annotated program of the form

$$
\begin{array}{l}
\{A\} \\
\textbf{while } BE \\
\quad \textbf{do } \{A \wedge BE\} \\
\quad P \\
\quad \{A\} \\
\textbf{enddo} \\
\{A \wedge \neg BE\}
\end{array}
$$

is complete if the annotated program

$$
\begin{array}{l}
\{A \wedge BE\} \\
P \\
\{A\}
\end{array}
$$

is complete.

For instance, the annotated program

$$
\begin{array}{l}
\{n = 5 \wedge i = 0 \wedge p = 1\} \\
\{n = 5 \wedge i \leq n \wedge p = 2^i\} \\
\textbf{while } i < n \textbf{ do } \{n = 5 \wedge i \leq n \wedge p = 2^i \wedge i < n\} \\
\quad \{n = 5 \wedge p = 2^i \wedge i < n\} \\
\quad \{n = 5 \wedge i + 1 \leq n \wedge p \times 2 = 2^{i+1}\} \\
\quad i \leftarrow i + 1; \\
\quad \{n = 5 \wedge i \leq n \wedge p \times 2 = 2^i\} \\
\quad p \leftarrow p \times 2 \\
\quad \{n = 5 \wedge i \leq n \wedge p = 2^i\} \\
\textbf{enddo} \\
\{n = 5 \wedge i \leq n \wedge p = 2^i \wedge i \geq n\} \\
\{n = 5 \wedge i = n \wedge p = 2^i\} \\
\{i = 5 \wedge p = 2^i\} \\
\{p = 32\}
\end{array}
$$

is complete. It justifies the formula

$$
\begin{array}{l}
\{n = 5 \wedge i = 0 \wedge p = 1\} \\
\textbf{while } i < n \textbf{ do} \\
\quad i \leftarrow i + 1; \\
\quad p \leftarrow x \times 2 \\
\textbf{enddo} \\
\{x = 32\}.
\end{array}
$$